Feature-Aware Test Generation for Deep Learning Models

XINGCHENG CHEN, Technical University of Munich, Germany and fortiss GmbH, Germany
OLIVER WEISSL, Technical University of Munich, Germany and fortiss GmbH, Germany
ANDREA STOCCO, Technical University of Munich, Germany and fortiss GmbH, Germany

As deep learning models are widely used in software systems, test generation plays a crucial role in assessing the quality of such models before deployment. To date, the most advanced test generators rely on generative AI to synthesize inputs; however, these approaches remain limited in providing semantic insight into the causes of misbehaviours and in offering fine-grained semantic controllability over the generated inputs. In this paper, we introduce DETECT, a feature-aware test generation framework for vision-based deep learning (DL) models that systematically generates inputs by perturbing disentangled semantic attributes within the latent space. DETECT perturbs individual latent features in a controlled way and observes how these changes affect the model's output. Through this process, it identifies which features lead to behavior shifts and uses a vision-language model for semantic attribution. By distinguishing between task-relevant and irrelevant features, DETECT applies feature-aware perturbations targeted for both generalization and robustness.

Empirical results across image classification and detection tasks show that DETECT generates high-quality test cases with fine-grained control, reveals distinct shortcut behaviors across model architectures (convolutional and transformer-based), and bugs that are not captured by accuracy metrics. Specifically, DETECT outperforms a state-of-the-art test generator in decision boundary discovery and a leading spurious feature localization method in identifying robustness failures. Our findings show that fully fine-tuned convolutional models are prone to overfitting on localized cues, such as co-occurring visual traits, while weakly supervised transformers tend to rely on global features, such as environmental variances. These findings highlight the value of interpretable and feature-aware testing in improving DL model reliability.

1 INTRODUCTION

Deep learning models such as deep neural networks (DNNs) and transformers have achieved impressive performance on many computer vision benchmarks [10], and they are widely deployed in many software systems nowadays. However, this success often fails to translate to real-world deployments, where models encounter inputs that differ, sometimes subtly, from those seen during training [37, 53]. These failures arise from two main challenges, namely *limited generalization* to unseen but valid inputs, and *lack of robustness* to spurious correlations that the model mistakenly relies on. Conceptually, these problems are related to the empirical risk minimization paradigm [47], which encourages DL models to optimize average performance over the training distribution, regardless of whether the features they exploit are semantically meaningful. As a result, models often learn *shortcut solutions*—spurious correlations

Authors' addresses: Xingcheng Chen, xingcheng.chen@tum.de, Technical University of Munich, Germany, xchen@fortiss.org and fortiss GmbH, Germany; Oliver Weißl, o.weissl@tum.de, Technical University of Munich, Germany, weissl@fortiss.org and fortiss GmbH, Germany; Andrea Stocco, andrea.stocco@tum.de, Technical University of Munich, Germany, stocco@fortiss.org and fortiss GmbH, Germany.

tied to background, lighting, pose, or co-occurring objects—that work on the training set but fail when exposed to real-world data distributions [13, 16].

While deriving from the same root problem, in literature, these two challenges have been investigated separately. While test generation techniques aim to evaluate model generalization by synthesizing new inputs that may uncover erroneous behaviors [36, 54], they often lack explainability and perturbed multiple features in an entangled manner (e.g., by injecting global Gaussian noise or simultaneously modifying lighting, texture, and object semantics). Therefore, generated tests uncover failures, but they do not clarify which feature changes caused the failure, or whether those changes are semantically relevant to the task. Conversely, *spurious feature analysis* focuses on *robustness*. They investigate whether models rely on invariant, task-irrelevant features, but typically through reactive post-hoc explanation tools such as saliency maps [6, 41]. This separation between input synthesis and feature diagnosis limits effective test generation and DL model improvements.

To fill this gap, in this paper, we propose DETECT, an explainable test generation framework that links *test generation* and *spurious feature identification* by systematically testing DNN behaviors under *semantically controlled perturbations*. Our approach is based on the insight that both generalization and robustness concerns can be addressed through *feature-aware changes* in the disentangled latent space: by precisely manipulating individual semantic features, either relevant or irrelevant to the task, DETECT assesses not only if the DNN model generalizes to novel inputs by varying the relevant features, but also whether it consistently misbehaves to irrelevant ones.

To achieve this, DETECT leverages a style-based generative model trained on the same distribution as the system under test (SUT), which provides access to a disentangled latent space of semantic features. By perturbing latent dimensions, we generate realistic input variants with fine-grained changes in specific attributes. This enables *feature-aware test generation*, by perturbing task-relevant features to explore decision boundaries to uncover untested regions or modifying task-irrelevant features to assess prediction invariance and identify spurious dependencies. DETECT incorporates a *feature-aware test oracle*, which quantifies model response to controlled perturbations using a logit-based criterion. If the model's prediction significantly shifts after modifying a semantically irrelevant feature, it is flagged as a robustness failure due to spurious correlation. Conversely, changes in response to relevant features are used to test generalizability.

In our empirical evaluation on image classification and detection tasks, DETECT outperforms existing test generation and spurious feature localization methods both in revealing feature-sensitive failures, as well as robustness failures due to task-irrelevant features. DETECT reveals spurious correlations in high-accuracy models that standard accuracy metrics overlook, and distinguishes between different shortcut behaviors learned by weakly and fully supervised models. On fine-grained tasks, DETECT also serves as an effective general test generator, offering flexible applicability across model types and data domains.

Our contributions are as follows:

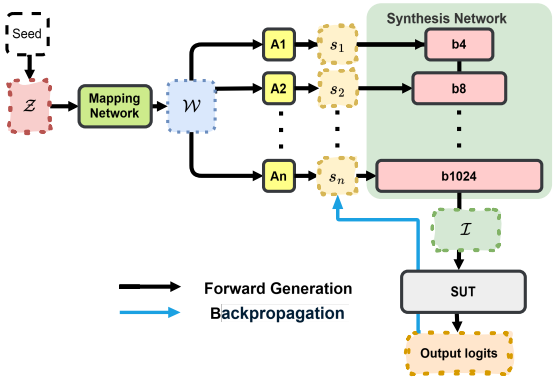

Fig. 1. **System Architecture of DETECT**: It comprises two main components, a StyleGAN generator and the SUT, and operates across three domains: the latent $\mathcal{S}$-space, the image space, and the output space. It contains 1) a forward generation pass from an initial seed z mapped into a set of style vectors s, which synthesize an image input passed to the SUT to obtain output logits; and 2) a backpropagation through SUT and synthesis network back to s.

1) We propose DETECT, a feature-aware testing framework that perturbs semantically meaningful latent features to evaluate both generalization and robustness;
2) We introduce a feature-aware test oracle that dually checks *invariance* for irrelevant features (robustness) and explores *decision boundaries* for relevant features (generalization);
3) We show that test generation and spurious feature analysis, though distinct in goal, can be combined into a unified methodology for probing, explaining, and repairing DNN defects.

2 BACKGROUND

In the context of deep learning testing, generative AI techniques are used due to their ability to model complex, feature-rich data distributions and synthesize new functional inputs through latent space manipulation [11, 30, 40, 49]. Among these, Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and diffusion models are widely used for image generation [30]. However, each comes with limitations for test generation purposes. VAEs, while structurally interpretable, tend to produce low-fidelity and therefore low quality input images. Conventional GANs suffer from entangled latent spaces, making semantic control over generated features difficult. Diffusion models can generate high-quality outputs but are computationally expensive, and also lack disentanglement in their latent representations.

Among these, recent approaches based on style-based GANs (StyleGAN) [21–23], offer high controllability due to disentangled and interpretable latent spaces. These models enable precise control over high-level image characteristics, referred to as *features*, such as texture, pose, lighting, or background, through separate style and content layers [49]. This controllability makes StyleGAN particularly well-suited

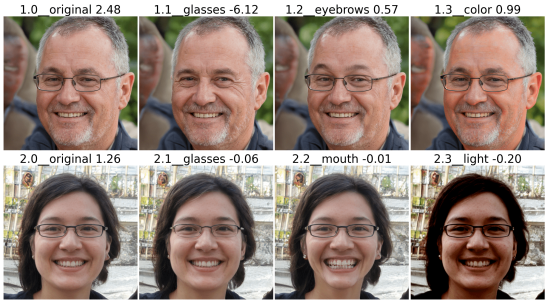

Fig. 2. Example perturbations in StyleSpace for glasses classification. Each row shows a base image (left) and variants generated by perturbing a single feature (i.e., glasses, eyebrows, skin color). Titles: channel index, feature label, model output logit.

for feature-aware test generation, where the goal is to manipulate specific semantic attributes of an image. As such, the remainder of this section briefly details the StyleGAN architecture.

StyleGAN [21–23] comprises two main components (see top left of Fig. 1): a mapping network and a synthesis network. The generation process begins with a randomly sampled latent code $z \in \mathcal{Z}$, which the mapping network then transforms into an intermediate latent space $\mathcal{W}$, that better captures disentangled image features. Subsequently, the synthesis network applies learned affine transformations to produce per-layer style vectors s, resulting in the StyleSpace ($\mathcal{S}$). These style vectors modulate the convolutional activations at each block, allowing the generator to control image features at different semantic and spatial levels—early (coarse) layers influence global properties (e.g., pose, shape), while later (fine) layers affect localized details (e.g., texture, lighting). Each style vector in $\mathcal{S}$ consists of multiple *channels*, where each channel controls a distinct and semantically meaningful visual attribute, thereby enabling precise and localized image manipulations. This property allows us to trace, perturb, and interpret the influence of individual style channels on downstream predictions.

Fig. 2 shows several examples related to the glasses detection task. By perturbing task-relevant features, one can control the presence or absence of glasses, resulting in a true negative prediction by the classifier on image 1.1 (where glasses are absent), but also a false negative prediction on image 2.1 (where instead glasses are present). Modifications on task-irrelevant features–such as eyebrows, facial expression, or sunlight–cause a drop in classifier confidence (such as images 1.2 and 1.3), and in some cases, even result in misclassifications (such as images 2.2 and 2.3). These findings highlight a critical challenge: not all influential features are task-relevant.

3 FEATURE-AWARE TEST ORACLE

A central component of our testing framework is a *feature-aware test oracle* that defines behavioral expectations based on the semantic relevance of input features. Unlike traditional test oracles that apply a uniform correctness criterion (e.g., *misclassification*)

across all inputs, our oracle distinguishes between *task-relevant* and *irrelevant* features, assigning distinct testing objectives to each category. First, we define a feature as a semantically interpretable factor in the latent space, whose controlled manipulation leads to a measurable change in the generated input.

Formalization. Let $\mathcal{T}$ be a K-class classification task, and let $\mathbf{F}(x) \in \mathbb{R}^K$ be the model's output logits for input x, with $\mathbf{F}_t(x)$ representing the logit for the target class $t \in \{1, \ldots, K\}$. Let $\phi \in \mathcal{F}$ be a controllable semantic feature of the input, manipulated through a generative model into a new input x'.

Definition 1 (Task-Relevant Feature). *A feature ϕ is **task-relevant** with respect to task $\mathcal{T}$ (denoted $\phi \in \mathcal{F}_{rel}^{\mathcal{T}}$) if perturbing ϕ induces a meaningful change in the ground truth label or alters the semantics necessary for correct prediction under $\mathcal{T}$. Let $x \sim_\phi x'$ denote that x and x' differ only in feature ϕ. Then, a feature ϕ is task-relevant if:*

$$\exists\,(x, x') \text{ such that } x \sim_\phi x', \quad label(x) \neq label(x'). \tag{1}$$

Definition 2 (Task-Irrelevant Feature). *A feature ϕ is **task-irrelevant** (denoted $\phi \in \mathcal{F}_{irr}^{\mathcal{T}}$) if changes in ϕ do not alter the ground true label under task $\mathcal{T}$, and are semantically non-essential to the prediction objective. Formally,*

$$\forall\,(x, x') \text{ such that } x \sim_\phi x', \quad label(x) = label(x'). \tag{2}$$

Definition 3 (Spurious Feature). *A feature $\phi \in \mathcal{F}_{irr}^{\mathcal{T}}$ is **spurious** if there exists a pair of inputs that differ only in ϕ, and this change exercises a significant influence on the model's prediction. Formally, ϕ is spurious if:*

$$\exists\,(x, x') \text{ such that } x \sim_\phi x', \quad \Delta_{logit}(\phi) = |\mathbf{F}_t(x') - \mathbf{F}_t(x)| > \tau, \tag{3}$$

where τ is a predefined confidence threshold.

Our oracle supports two complementary testing goals:

Oracle misclassification. It applies to relevant features for behaviour exploration (i.e., DNN generalization). For $\phi \in \mathcal{F}_{\text{rel}}^{\mathcal{T}}$, we perturb inputs to explore semantic behaviours. Prediction changes here are expected and indicative of the model's sensitivity to meaningful feature variation. Misclassifications near these transitions signal overfitting or under-generalization.

Oracle confidence. It applies to irrelevant features for DNN robustness assessment. For $\phi \in \mathcal{F}_{\text{irr}}^{\mathcal{T}}$, the model should behave invariantly. Thus, this oracle monitors $\Delta_{\text{logit}}(\phi)$ as a proxy for overreliance on non-causal features. Large shifts indicate spurious behavior.

In practice, due to the hierarchical and interdependent nature of some features, the latent space of the generator cannot be perfectly disentangled. We select $\mathcal{S}$ space as the manipulation space, as it achieves a disentanglement score of 0.75, much higher than 0.54 measured from $\mathcal{W}$ space [51]. In this sense, $\mathcal{S}$ space represents the most disentangled and controllable space currently attainable for our purpose. To ensure methodological rigor, we adopt a conservative attribution rule: when a feature modification introduces a clear relevant semantic change but also causes small

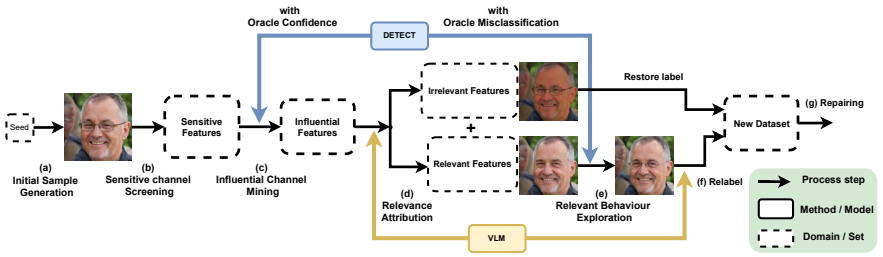

Fig. 3. Sequential Pipeline of DETECT.

incidental adjustments, we treat the feature as relevant and ignore such minor co-effects. For example, modifying the presence of glasses while slightly affecting nearby shading or eyebrow shape.

4 APPROACH

Fig. 1 illustrates the workflow of our proposed framework, which we call DETECT. DETECT first samples a random latent seed, which is passed through a StyleGAN trained on the same domain as the SUT to generate an image (step Initial Sample Generation). To understand which channels in each latent feature s_i affect the model's prediction, DETECT estimates the sensitivity of each channel on the target logit with explainable AI (XAI) methods (step Sensitive Channel Screening). Then, DETECT invokes an oracle-aware perturbation loop twice with different oracles. In the first iteration (step Influential Channel Mining), DETECT perturbs all sensitive features along the direction that reduces the SUT's confidence. and collect the *influential* features that cause significant output changes in the model's output. DETECT then identifies whether these are task-relevant using a Vision Language Model (VLM) (step Relevant Attribution). If a feature is found to be semantically irrelevant yet influences the output, DETECT labels it as *spurious*. On the other hand, for features deemed relevant, DETECT performs a second perturbation guided by the Oracle Misclassification to explore the SUT behaviours (step Relevant Behavior Exploration). The second perturbation is optimized via hill climbing until decision-boundary tests are obtained. These new inputs are then relabeled by the VLM, producing a test suite that captures both relevant and spurious feature variations.

In the remainder of the section, we describe each step in greater detail.

4.1 Initial Sample Generation

First, a random latent seed z is generated and passed through a StyleGAN mapping network to obtain an intermediate latent vector $w \in \mathcal{W}$ (Fig. 1). This vector is then transformed into a set of style vectors s_i, each of which modulates a specific synthesis block and spatial resolution during image generation. Then, the generated image $\mathcal{I}$ is fed into the SUT to obtain class logits. DETECT uses the logit output of the target class because it provides a direct, unnormalized measure of the SUT's confidence, allowing

precise assessment of the model's sensitivity to controlled changes in specific latent features.

4.2 Sensitivity Screening Methods

Unlike most search-based latent space test generation methods exploring the entire high-dimensional feature space [30, 49] (typically spanning thousands of latent dimensions), DETECT employs XAI-based sensitivity estimation to efficiently screen sensitive features and directions, reducing the subsequent perturbation search space.

We define the sensitivity of the SUT's target logit $y[t]$ to the i-th style vector s_i, where $\mathbf{G}(s)$ denotes the synthesis network that synthesizes the image from the style vector $s = \{s_1, ..., s_n\}$ and $s_i \in \mathbb{R}^d$ (the length of style vector d varies among different layers), i.e.,

$$\alpha_i^{(t)} = \mathbf{S}\left(y[t], s_i\right) = \mathbf{S}\left((\mathbf{F}_t \circ \mathbf{G})(s), s_i\right) \in \mathbb{R}^d, \tag{4}$$

where $y[t]$ denotes the logits of SUT for the target class t and the sensitivity α_i that has the same dimension as s_i.

In this work, we adopt three established XAI methods, applied not to the image space, but to the disentangled latent space.

Gradient Saliency [42]. It computes the sensitivity of the target logit $y[t]$ with respect to each style vector s_i by backpropagation through the composed function $\mathbf{F}_t \circ \mathbf{G}$, i.e.,

$$\alpha_i^{(t)} = \frac{\partial y[t]}{\partial s_i} = \frac{\partial (\mathbf{F}_t \circ \mathbf{G})(s)}{\partial s_i}. \tag{5}$$

This serves as a first-order approximation of the local sensitivity of the output to each style component.

SmoothGrad [43]. While pure gradients are often noisy and sensitive to local variations in the input space, we adopt SmoothGrad to improve stability by averaging gradients over noise-perturbed input. We sample N noisy $\{s^{(j)} = s + \epsilon^{(j)}\}_{j=1}^{N}$ with $\epsilon^{(j)} \sim \mathcal{N}(0, \sigma^2 I_d)$, and then estimate for the sensitivity of channel s_i to output logit $y[t]$ as:

$$\alpha_i^{(t)} = \frac{1}{N} \sum_{j=1}^{N} \frac{\partial y[t]}{\partial s_i^{(j)}} = \frac{1}{N} \sum_{j=1}^{N} \frac{\partial (\mathbf{F}_t \circ \mathbf{G})(s^{(j)})}{\partial s_i^{(j)}}. \tag{6}$$

Finite-Difference Approximation [32]. This quantifies the local effect of each latent channel via output variation by computing a gradient-free sensitivity approximation based on finite differences. This method is applicable even when the composed model $\mathbf{F} \circ \mathbf{G}$ is non-differentiable or analytically inaccessible. For each channel $s_{i,c}$ of style vector s_i, we apply a small perturbation Δ and the sensitivity of channel and estimate sensitivity as:

$$\alpha_{i,c}^{(t)} = \frac{\mathbf{F}_t \circ \mathbf{G}(s_i + \Delta \cdot e_c) - \mathbf{F}_t \circ \mathbf{G}(s)}{\Delta}, \tag{7}$$

where e_c is the c-th standard basis vector in $\mathbb{R}^d$.

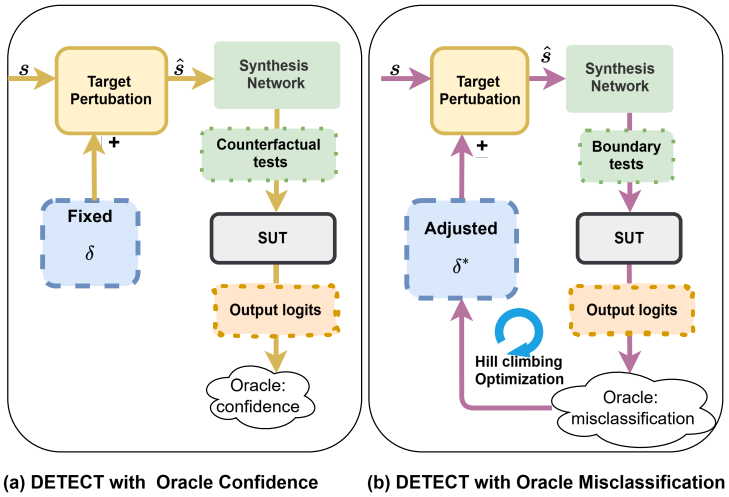

Fig. 4. DETECT with Oracle Confidence and Misclassification.

After computing the sensitivity scores, DETECT retrieves a channel-wise attribution map for each layer in the synthesis network. Since each style vector can have up to a thousand latent channels, only the top-k channels with the highest absolute sensitivity scores are retained for further analysis. Channels outside this set are considered inactive or behaviorally negligible with respect to the SUT. While we use absolute sensitivity for ranking, we preserve the sign of each score. In fact, the directionality of the attribution (i.e., whether increasing/decreasing a channel pushes the output logit up or down) is used by the perturbation procedure to apply signed directional changes to the most influential channels.

4.3 Perturbation Loop

Once sensitive channels have been identified, DETECT invokes oracle-specified perturbations to evaluate their behavioral influence, as outlined in stages (c) Influential Channel Mining and (e) Relevant Behaviour Exploration in Fig. 3. This two-stage perturbation is guided by the feature-aware oracles described in Section 3. Fig. 4 provides an overview of these two configurations. In the first configuration (a), the perturbation δ is fixed, and the Oracle Confidence guides the identification of influential features by measuring the change in output logits. In the second configuration (b), the perturbation δ is iteratively adjusted through hill-climbing optimization, using misclassification as the oracle, to generate tests located near the decision boundary. The detailed algorithmic steps of this loop are provided in Algorithm 1.

Algorithm 1 illustrates the test generation by perturbing latent channels under a specified oracle. Here, C denotes a set of selected channels (identified via sensitivity analysis or semantic relevance), and O specifies the behavioral oracle, either confidence or misclassification.

Algorithm 1 ChannelPerturb$(s, \mathbf{F}, C, O, \tau)$

Require: Style vector s, classifier $\mathbf{F}$, channel set C, oracle O, threshold τ
1: Initialize perturbed result set $\mathcal{R} \leftarrow \emptyset$
2: **for** each (l, c) in C **do**
3: Perturb $\hat{s}_l[c] = s_l[c] + \delta$
4: Generate $\hat{x} = \mathbf{G}(\hat{s})$, $\hat{y} = \mathbf{F}(\hat{x})$
5: **if** O = `confidence` and $\Delta_{confidence} > \tau$ **then**
6: Add $(l, c, \hat{x})$ to $\mathcal{R}$
7: **else if** O = `misclassification` and $\mathcal{L}(\hat{y}) \neq \mathcal{L}(y)$ **then**
8: Refine δ^* via hill climbing to decision boundary
9: Add $(l, c, \hat{x}^*)$ to $\mathcal{R}$
10: **end if**
11: **end for**
12: **return** $\mathcal{R}$

The main loop of the algorithm enumerates each channel $(l, c) \in C$, and then applies a perturbation δ on each selected channel $s_{l,c}$ scaled by a fixed step size ϵ (Line 3). The step size ϵ is chosen to induce a visible semantic change in the generated image, while avoiding artifacts or unrealistic distortions [30, 51]. The step size ϵ is determined by the scale of the latent space, independent from the dataset or the downstream SUT, since the values in $\mathcal{S}$ space after affine transforms lie in stable numeric ranges [22]. The perturbation step δ is constructed to reduce the model's confidence in the current prediction. Its formulation differs slightly between binary and multiclass classification, as shown below:

$$\delta = \begin{cases} -\epsilon \cdot \text{sign}(\alpha_{i,c}^{(t)}) \cdot \text{sign}(y[t]), & \text{binary,} \\ -\epsilon \cdot \text{sign}(\alpha_{i,c}^{(t)}), & \text{multiclass.} \end{cases} \tag{8}$$

This design ensures that the perturbation moves in the direction opposite to the gradient-derived influence, optionally adjusting for logit polarity in binary classification.

The perturbed latent $\hat{s}$ is then used for synthesizing the perturbed image $\hat{x} = \mathbf{G}(\hat{s})$, and obtain the system's response $\hat{y} = \mathbf{F}(\hat{x})$ (Line 4). The logit $\hat{y}$ is analyzed according to the selected oracle. Under the confidence-based oracle, Detect records the perturbed image if the drop of confidence in the target logit exceeds a predefined threshold, indicating SUT's prediction relies on the perturbed feature (Lines 5-6). Under the misclassification-based oracle, Detect checks whether the predicted label changes (Lines 7-9). We assume $\mathcal{L}(\cdot)$ returns the predicted class. For binary classification, $\mathcal{L}(\cdot)$ checks a sign in the single logit; for multiclass, it takes the top-1 class. If misclassification occurs, Detect refines the perturbation δ using a hill-climbing strategy to find the minimal modification required to cross the decision boundary, thereby producing more precise boundary examples $\hat{x}^*$.

Fig. 5. Example of Image Input to VLM.

Algorithm 1 is executed twice: first, to probe all sensitive channels using the confidence-drop oracle; and second, only on task-relevant candidates to explore the model's behavior boundaries using the misclassification oracle. Specifically, DETECT first obtains channel-wise sensitivities using one of the methods in Section 4.2 and then extracts the top-k most sensitive channels c by sorting their absolute sensitivity scores to form a candidate set $C_{sensitivity}$. Then we apply the first-stage CHANNELPERTURB with the confidence-based oracle to isolate latent channels whose perturbation causes significant behavioral shifts. These are referred to as *influential* channels. However, at this point, DETECT does not yet distinguish whether the altered features are task-relevant or irrelevant, which is done in the last step.

4.4 Relevance Attribution, Relabel, and Repair

In this section, DETECT incorporates a vision-language model (VLM) both as a semantic evaluator and as an automated annotator, as illustrated in steps (d) Relevance Attribution and (f) Relabel in Fig. 3. The VLM provides textual-aligned supervision for generated images, enabling DETECT to automatically determine whether each perturbation expresses a task-relevant concept or a spurious one.

DETECT's relevance attribution module systematically categorizes each feature ϕ into $\mathcal{F}_{\text{rel}}^{\mathcal{T}}$ or $\mathcal{F}_{\text{irr}}^{\mathcal{T}}$ in a data-driven manner. Specifically, as an example shown in Fig. 5, the image input to VLM is composed of three pictures spliced together: (1) the original image, (2) the image after a single latent-direction change, and (3) a difference mask that highlights where the change occurs. The prompt instructs the VLM only to judge changes to the relevant attribute. For each channel in the influential feature set, DETECT aggregates VLM decisions across multiple counterfactual samples and assigns a label by majority voting. Channels whose perturbations align with the task semantics are labeled as relevant, while those inducing visually noticeable but task-irrelevant changes are labeled as spurious. For instance, in Fig. 5, a perturbation majorly alters the eyebrow region without affecting the presence of glasses; such a change is identified as task-irrelevant.

Beyond this semantic supervision, DETECT also exploits the inherent spatial hierarchy of StyleGAN. Empirically, channels from coarse-to-middle layers—responsible for high-level attributes such as shape, pose, or object identity—tend to correspond to relevant semantics, whereas fine-layer perturbations often lead to superficial changes like lighting or color shifts [22].

Then, for the subset of relevant features, the VLM is employed again to relabel the decision-boundary images produced in the oracle misclassification loop, since the original labels are no longer preserved. For spurious features, DETECT restores the original labels, consistent with our assumption of feature invariance.

Finally, DETECT incorporated both relevant and spurious features into a repaired dataset for fine-tuning of SUT to improve both generalizability and robustness. The fine-tuning follows a lightweight strategy using a small learning rate and limited epochs, allowing the model to adapt to corrected feature-label associations without overfitting to the generated data.

5 EMPIRICAL STUDY

5.1 Research Questions

RQ_1 (configuration): What configuration of DETECT yields the most effective and efficient feature-aware testing?

$RQ_{1.1}$: Which sensitivity screening strategy best identifies impactful latent channels?

$RQ_{1.2}$: What is the optimal number of channels to balance coverage and efficiency?

$RQ_{1.3}$: How do different VLMs affect the accuracy and cost-efficiency of feature attribution and labeling?

RQ_2 (behaviour exploration): How effective is DETECT at finding misbehaviors?

RQ_3 (spurious feature detection): How effective is DETECT at discovering failures due to spurious features?

RQ_4 (model repairing): Does the generated test cases contribute to improving the SUT?

RQ_1 investigates how to best configure DETECT for effective and efficient feature-aware testing. We examine which sensitivity screening strategy most effectively identifies impactful latent channels for targeted perturbation, explore how many channels should be perturbed to balance coverage and efficiency, and analyze how different VLMs influence the accuracy and cost-efficiency of feature attribution and labeling. RQ_2 evaluates DETECT's capability to uncover behavioral misbehaviors in the system under test (SUT). RQ_3 focuses on assessing whether DETECT can reveal failures caused by spurious or non-robust features. Finally, RQ_4 investigates whether the generated test cases can be leveraged to improve the SUT through retraining.

5.2 Metrics used for Analysis

For $RQ_{1.1}$, we compare different configurations of DETECT in terms of their runtime (in seconds) and the extent to which they alter the SUT's behavior.

For $RQ_{1.2}$, we measure the number of features in each layer that can affect the behavior of the SUT.

For $RQ_{1.3}$, we measure the agreement rate between the VLM and multiple human annotators. Besides, to mitigate inconsistency among different annotators, we evaluate consensus agreement against a majority-vote human label, where aggregated labels from multiple annotators represent the consensus ground truth. In addition to

correctness, we also assess the monetary cost per evaluation and average response time to measure their computational and economic efficiency.

For RQ_2, we evaluate the runtime of the compared methods in seconds. To assess whether structural information is preserved in the generation, we compute the MS-SSIM [48] score, which extends SSIM by computing similarity across multiple scales via iterative downsampling. This multi-scale structure is more in line with human perception than regular SSIM. Since MS-SSIM is less sensitive to color variation, we complement it with the normalized $\widetilde{d}_2$-Image distance defined as $\widetilde{d}_2(x_i, x_j) = L_2(x_i - x_j)/L_2(\mathbf{1}^x)$, where $\mathbf{1}^x$ is an all-ones matrix with the same shape as x. This metric aims to capture the overall magnitude of change, including color shifts. Finally, we evaluate the proximity to the decision boundary to assess the effectiveness of the generated test cases in challenging the model's predictions.

For RQ_3, we propose a simple metric to quantify how much a model relies on task-relevant features, i.e., the proportion of influential channels that are task-relevant:

$$R_{Relevance} = \frac{|\text{Task-Relevant Influential Features}|}{|\text{All Influential Features}|}.$$

This metric captures the alignment between a model's decision-making process and the task-defined semantics. A higher value indicates that the model's behavior is driven more by meaningful features, whereas a lower value suggests susceptibility to spurious correlations.

5.3 Objects of Study

5.3.1 Datasets and Models. To ensure the generality and robustness of our evaluation, we selected three diverse feature-rich datasets, each paired with a corresponding SUT.

Face Attributes (CelebA [29]). This dataset is labeled with 40 facial attributes, with each attribute as an independent binary classification task, although our subsequent analyses primarily focus on the eyeglasses and gender attributes. We evaluate two different SUTs: (1) *ResNet50 [17]*, consisting of a pretrained backbone followed by a custom fully connected classification layer. The entire model is fully fine-tuned on CelebA to predict all 40 attributes. It achieves 99.70% test accuracy on the eyeglasses attribute and 98.17% on gender. (2) *SWAG ViT [31]* a Vision Transformer with a frozen pretrained backbone and a custom two-layer classification head, trained in a weakly supervised manner on CelebA attributes. This model is chosen for its reported robustness against spurious features. It achieves a test accuracy of 99.17% on the eyeglasses attribute and 98.91% on the gender attribute.

Dog breed Classification (Dogs [26]). We use a publicly available dataset from Kaggle [26], which includes labeled images of 78 dog breeds. This dataset is used to evaluate multi-class classification performance. Only dog images were retained for our multi-class classification task. The model under test is a pretrained ReXNet-150 [15], which achieved a test accuracy of 87.9%.

Car Detection (COCO [27]). We evaluate object detection performance on the car class. The supervised model under test is YOLOv8, implemented via the official Ultralytics framework [19]. The model is pretrained on the COCO dataset, and we directly

apply this pretrained model to detect cars in our images without any fine-tuning. On the LSUN Cars dataset [52], we report an accuracy of 97.63% when including trucks, since some vehicles are ambiguous and may be classified as either car or truck; accuracy considering only cars is 78.83%.

5.3.2 Generator and VLM. Each dataset is paired with a pretrained StyleGAN generator and segmentation model. We use the official StyleGAN2 models pretrained on FFHQ [23], LSUN Dog [52], and LSUN Car [52], and further fine-tune each generator on the corresponding downstream dataset using a few thousand task-specific images (∼5k steps) and adaptive discriminator augmentation (ADA). This setup follows common practice in latent-space test generation methods [30, 33], where pretrained generators are preferred because their latent spaces capture a wider range of features learned from large and diverse datasets. Directly training a generator from scratch on the same dataset as SUT typically limits the richness of the learned feature space.

5.3.3 Configurations. Regarding configuration of sensitive channel screening ($RQ_{1.1}$), we evaluate three methods in DETECT, including two white-box approaches, *Gradient Saliency* (*Grad*) and *SmoothGrad*, and one black-box approach, the *Finite-Difference Approximation* (*FDA*). We chose these methods for different application requirements and computational complexity. *Grad* requires a single backward pass through the SUT and generator. *SmoothGrad* increases this cost by performing N backward passes, each on a perturbed version of the input, to generate more robust attributions through averaging. Following prior work [43], we set $N = 10$. In contrast, *FDA* is gradient-free and suitable for black-box settings. Instead, it perturbs each individual channel of style vectors and observes the resulting changes. Given that our StyleGAN model contains $9,088$ channels, *FDA* performs $9,088$ SUT forward passes to approximate relevance. Besides, to support relevance attribution and relabel procedures ($RQ_{1.3}$), we evaluate 8 VLMs covering different model families, sizes, and reasoning capabilities. The selected models include Qwen-VL [7] (8B-Thinking, 30B-A3B-Thinking, and Max), Mistral VLM variants [18] (Small-3.1-24b-instruct, Medium-3.1), and OpenAI models [2] (GPT-o4-Mini, GPT-4.1).

Regarding RQ_2, to ensure a fair comparison with conventional test generation methods, we also define a simplified version of our framework, denoted as DETECT*. Unlike the full pipeline, DETECT* omits the relevance attribution step and perturbs all channels identified as sensitive in the initial screening. It relies solely on the misclassification-based oracle and thus functions as a general-purpose test generator, targeting exposing misbehaviours under small, single-feature perturbations. This configuration is used when semantic relevance cannot be reliably established. In certain domains, relevance attribution requires expert-level domain knowledge and cannot be handled reliably by general-purpose VLMs, such as the medical applications or fine-grained species identification (*Dogs*). In other settings, the generator may be trained exclusively on a specific domain, so all latent variations describe within-class changes, such as *COCO-Cars*. In such cases, distinguishing relevant from spurious features becomes unnecessary since all the detected features are deemed irrelevant. In

such cases, DETECT* serves as a domain-agnostic variant that allows fair comparison with conventional latent-space test generators.

5.3.4 Comparison Baselines. Although no existing method jointly addresses the dual objectives of our approach, we compare DETECT to representative baselines that align with each individual testing objective, latent-space test input generation, and spurious feature identification

Mimicry. A state-of-the-art latent-space test generation method, that exploits the $\mathcal{W}$-space of StyleGAN. It interpolates between w-latent vectors from different class instances and uses population-based optimization to tune the interpolation weights. The resulting synthesized images form test cases positioned near the SUT's decision boundary [49]. Like many other latent-space test generators, the entire latent vector is manipulated during this process, confounding relevant and irrelevant features. In contrast, DETECT operates in the more disentangled $\mathcal{S}$-space and efficiently screens sensitive features to the SUT and only perturbs a single semantic feature at a time. To support a fair comparison, we primarily compare Mimicry against DETECT*, the simplified variant of our framework that does not use relevance attribution and perturbs all channels identified as sensitive. Mimicry serves as a strong baseline for evaluating generated test cases, and has been shown to surpass earlier methods such as Sinvad [20] and DeepJanus [38] in boundary testing settings.

SpRAy (spectral relevance analysis). An eigenvalue-based clustering method for spurious feature localization proposed by Lapuschkin et al. [4, 25]. It is based on computing Layer-wise Relevance Propagation (LRP) heatmaps for model predictions and then applying spectral clustering to group similar explanation patterns. We include SpRAy as a baseline to evaluate effectiveness in spurious feature detection, since it remains one of the few techniques that directly target the automatic detection of spurious features. SpRAy represents the pixel-level explanation family of methods and clustering saliency explanations. As such, SpRAy serves as a representative and methodologically relevant baseline for assessing the effectiveness of spurious feature detection from explanation patterns, complementing our feature-perturbation-based analysis.

5.4 Procedure

In RQ_1, we apply each sensitive channel screening method using the confidence-based oracle on the facial task for 50 randomly selected seeds. This is justified as channel sensitivity mainly depends on StyleGAN's internal structure rather than task semantics, making the results broadly transferable and reused across datasets in RQ_2 and RQ_3. We fine-tuned the perturbation $\epsilon = 10$ to induce visible semantic changes and avoid artifacts, following existing work [30, 40]. Additionally, we analyze the contribution of specific synthesis layer channels to justify the selection of the number of candidate channels in subsequent experiments. To compute the human-aligned metrics for $RQ_{1.3}$, we collected human annotations using AWS Mechanical Turk [44]. Two tasks were evaluated: feature attribution and boundary-image relabeling. In total,

69 participants completed 1,069 annotation questions. We included several attention-check questions to detect inattentive responses and discarded 22 submissions that failed these checks.

In RQ_2, we select the best-performing configuration from RQ_1 against Mimicry. For this comparison, we use the oracle misclassification, which resembles the boundary-based objectives used in Mimicry [49] and evaluate with 50 seeds by a random number generator (RNG).Following existing guidelines in generative AI-based test generation [23, 30], Detect applies latent truncation to improve image quality. For the face and dogs tasks, we use a truncation value of $\psi = 0.7$, while for cars detection we use $\psi = 0.5$. Beyond runtime, we assess the visual impact of manipulations using d_2-distance (lower is better) and MS-SSIM [48] (higher indicates greater structural similarity).

In RQ_3, we evaluate spurious feature detection using 88 RNG seeds for both the ResNet and SWAG models on the eyeglasses classification task and 50 RNG seeds for YOLO on the car detection task. We use a 40% confidence drop threshold to define influential features in the confidence-based oracle, $\tau = 0.4|y[t]|$, to capture impactful shifts in prediction without overreacting to minor fluctuations. We re-implemented the SpRAy method based on the paper description [25], as no official code was provided. The procedure includes the following steps: First, we generate LRP relevance maps for positive samples in the test set. Then, we pre-process the maps by resizing and flattening them into uniform vectors and perform eigengap analysis on the k-nearest neighbor affinity graph to determine the optimal number of clusters. Finally, we apply spectral clustering [35] and visualize clusters via t-SNE [46]. This process serves as a baseline to assess the capability of Detect on identifying spurious correlations.

In RQ_4, we fine-tune the pretrained CelebA ResNet classifier using a small repaired dataset generated by Detect. More specifically, the repairing dataset mixes original CelebA training samples (80%) with generated counterfactuals (20%). Only the final classification layer is updated, while the backbone remains frozen. The model is optimized with a small learning rate ($\text{lr} = 2 \times 10^{-5}$), early stopping for at most 20 epochs, and a loss targeting only the finetuned attribute (eyeglasses), ensuring that the model adapts specifically to corrected feature–label associations. To ensure that evaluation is conducted on data unseen during fine-tuning, we create a held-out generated test set by reserving 20% of all VLM-relabeled counterfactuals using a fixed random seed.

5.5 Results

5.5.1 RQ_1 (configuration). Table 1 summarizes the average confidence dropand runtime for each sensitive channel screening method, along with their standard deviations. The two white-box methods, *Grad* and *SmoothGrad*, perform similarly in reducing model confidence, while *FDA* is less effective. In terms of runtime, *Grad* is the fastest, with *SmoothGrad* incurring only modest additional cost due to the multiple backward passes. *FDA*, by contrast, is slower due to its gradient-free sensitivity estimation.

To better understand how many sensitive channels are required for effective perturbation, we analyze the distribution of activated channels across StyleGAN's synthesis

Table 1. RQ_1 (configuration) results (**best** & second best).

	Grad	*SmoothGrad*	*FDA*
Confidence Drop ↑	0.506 ± 1.230	**0.581 ± 1.287**	0.062 ± 0.385
Runtime (sec) ↓	**5.486 ± 0.294**	5.838 ± 0.290	139.530 ± 1.164

layers in cases where the SUT exhibits a confidence drop greater than 1. This threshold removes low-signal noise and retains only semantically impactful perturbations, as it reflects a substantial logit change.

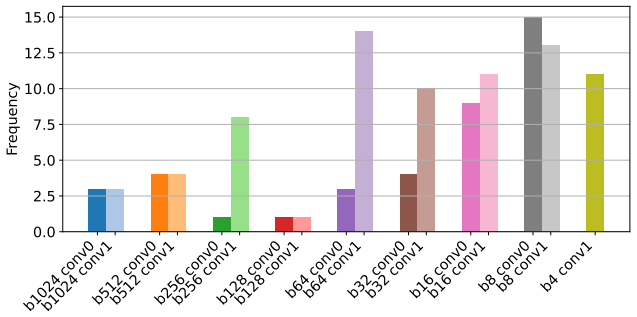

Fig. 6. RQ_1 (configuration): Number of channels with confidence drop greater than one for each StyleGAN layer in DETECT.

As shown in Fig. 6, only a subset of channels, mainly from early layers, are consistently involved in these impactful cases. This analysis suggests that a limited number of candidate channels is sufficient to induce meaningful behavioral shifts while maintaining efficiency. Specifically, we select the top-15 most sensitive channels from the coarse-to-middle layers, and the top-5 from the fine layers (beyond 512 resolution) for subsequent tasks. While earlier layers are known to control high-level semantics [22], our results confirm that their channels are disproportionately represented among those driving significant output changes.

Fig. 7 compares the performance of 7 VLMs across four dimensions. The top row shows their agreement with human annotators for the two tasks: Relevance Attribution and Relabeling. Each bar pair reports both the raw agreement and the majority-vote consensus agreement. The Relevance Attribution task is more challenging, where models such as qwen3-vl-8b-thinking and gpt-o4-mini achieve the highest alignment with human judgments. For the Relabeling task, performance is generally more stable: all models except mistral-small-3.1-24b-instruct attain high agreement levels close to the human upper bound. The bottom row presents the cost per query and generation time, revealing substantial differences in efficiency: lighter models incur minimal cost and latency, while larger models such as gpt-4.1 exhibit noticeably higher computational overhead. Considering both semantic alignment and efficiency trade-offs in

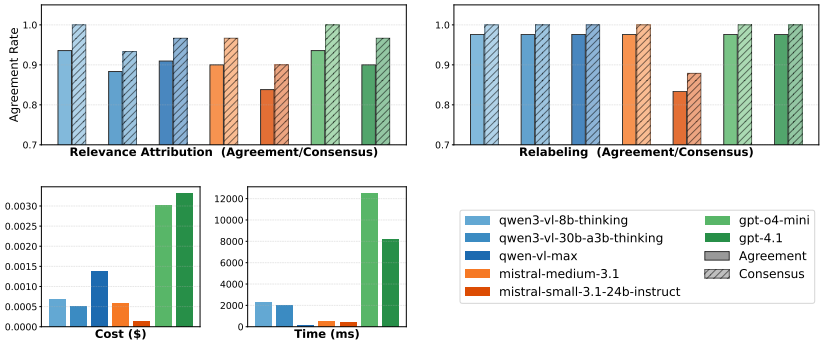

Fig. 7. RQ_1 (configuration): Performance of VLMs comparison.

cost and latency, we select qwen3-vl-8b-thinking as the VLM used in all subsequent experiments.

RQ_1 (configuration): *SmoothGrad offers the best overall trade-off between attribution quality and computational cost among sensitivity screening methods. Analysis of impactful perturbations reveals that selecting top-15 channels per early/middle layer and top-5 per fine layer balances effectiveness with efficiency and captures most impactful perturbations. Across both attribution and relabeling tasks, the evaluated VLMs show high alignment with human judgments, with qwen3-vl-8b-thinking providing the best balance of semantic accuracy and efficiency.*

5.5.2 RQ_2 (behavior exploration). Table 2 shows that both DETECT across all four tasks and three metrics—runtime, $\widetilde{d}_2$-Image distance, and MS-SSIM—DETECT* consistently outperforms MIMICRY, which is confirmed by a Mann–Whitney U test [50] ($\sigma > 2$ indicates significance) and Cohen's d effect sizes [8] (★for $d > 1.0$). For the d_2-Boundary metric, results are more mixed. Under d_2-Boundary, DETECT* outperforms the baseline on *CelebA*-gender and *CelebA*-glasses, with both DETECT variants surpassing MIMICRY on *CelebA*-glasses. Here, the better performance of DETECT* against MIMICRY is statistically significant. For multi-class tasks, MIMICRY performs better in *COCO-Cars*, while for *Dogs* we could not identify a clearly better approach.

We also provide visual comparison of the generated examples for DETECT and MIMICRY across all three tasks in Appendix A. These examples illustrate the qualitative differences between the two methods and support the quantitative trends reported above. In particular, DETECT generally produces more focused and attribute-specific edits, while MIMICRY often introduces broader or incidental changes.

Table 2. RQ_2: Effectiveness and efficiency results.

		Runtime (sec) ↓	$\widetilde{d}_2$-Image ↓	MS-SSIM ↑	d_2-Boundary ↓
CelebA glasses	DETECT	1.288 ± 0.354	**0.084 ± 0.025**	0.730 ± 0.054	**0.035 ± 0.051**
	DETECT*	**1.256 ± 0.384**	0.088 ± 0.043	**0.731 ± 0.053**	0.060 ± 0.141
	MIMICRY	96.795 ± 0.751	0.250 ± 0.049	0.420 ± 0.073	2.996 ± 0.434
	Significance	8.07σ ★	7.72σ ★	8.03σ ★	8.07σ ★
CelebA gender	DETECT	**1.277 ± 0.354**	**0.098 ± 0.025**	0.668 ± 0.090	2.442 ± 2.275
	DETECT*	3.846 ± 2.830	0.115 ± 0.083	**0.714 ± 0.087**	**0.566 ± 1.109**
	MIMICRY	96.342 ± 0.676	0.247 ± 0.046	0.415 ± 0.065	1.867 ± 0.624
	Significance	12.00σ ★	10.26σ ★	11.70σ ★	9.33σ ★
COCO Cars	DETECT*	**3.247 ± 2.208**	**0.094 ± 0.045**	**0.759 ± 0.072**	0.644 ± 0.038
	MIMICRY	50.982 ± 0.702	0.251 ± 0.044	0.332 ± 0.083	**0.259 ± 0.200**
	Significance	11.50σ ★	11.07σ ★	11.43σ ★	−5.85σ
Dogs	DETECT*	**3.697 ± 2.423**	**0.073 ± 0.043**	**0.819 ± 0.100**	0.457 ± 0.118
	MIMICRY	12.325 ± 0.674	0.281 ± 0.065	0.157 ± 0.134	**0.422 ± 0.157**
	Significance	12.13σ ★	11.82σ ★	11.72σ ★	−0.71σ

RQ_2 (behavior exploration): *DETECT produces more targeted and visually precise manipulations than MIMICRY while preserving structural similarity at a lower runtime, confirming a more controlled and efficient test generation.*

Table 3. RQ_3: Comparison of spurious feature detection between ResNet and SWAG models. Only frequently used channels (by > 1 input) are selected.

	ResNet	**SWAG**
Influential inputs (selected/ total)	415 / 548	168 / 524
Channels used (selected / total)	94 / 236	60 / 450
Relevant/Spurious inputs	329/86	55/113
Relevant/Spurious channels	68/36	39/56
$R_{Relevance}$	0.65	0.41

5.5.3 RQ_3 (spurious feature detection). Table 3 reports the results for spurious feature identification in glasses classification, distinct for a fully fine-tuned ResNet50 and a frozen-backbone SWAG ViT. Although both models achieve high test accuracy on the eyeglasses attribute, their behavior under controlled latent perturbations diverges substantially.

ResNet50 identifies more task-relevant influential inputs (329 vs. 55) and channels (68 vs. 39), whereas SWAG yields more spurious influential inputs (113 vs. 86) and channels (56 vs. 36). The relevance ratio $R_{\text{Relevance}}$ is higher for ResNet (0.65 vs. 0.41), indicating stronger alignment with semantically meaningful features. Despite SWAG's theoretical robustness [31], it shows greater reliance on spurious features. These

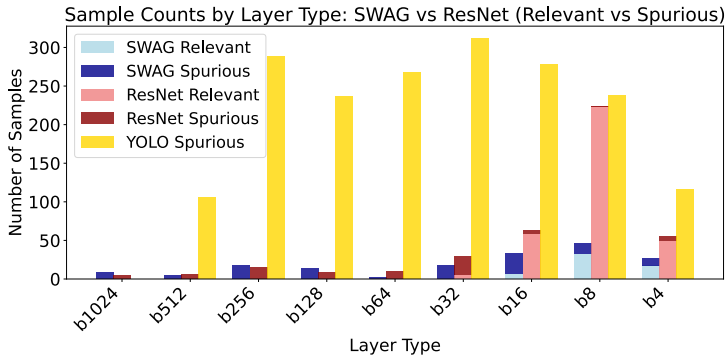

Fig. 8. RQ_3: distribution of relevant and spurious attributions across StyleGAN layers for ResNet and SWAG.

results suggest that SWAG's frozen backbone may limit task adaptation, increasing dependence on incidental cues.

To analyze these results more qualitatively, Fig. 8 shows the distribution of relevant and spurious features across StyleGAN layers. Relevant channels concentrate in coarse blocks (b4–b16), which encode semantic structures like shape. In contrast, spurious channels appear more often in mid-to-late layers (e.g., b512 and above), linked to texture and lighting, confirming that many spurious dependencies stem from low-level artifacts. While ResNet focuses on relevant channels, SWAG exhibits a more dispersed activation pattern with fewer relevant and more spurious attributions, reinforcing the risk of lacking adaptation to relevant features in a weakly supervised manner. Notably, both models' spurious activations concentrate in mid-to-late layers tied to lighting and background, underlining the value of layer-aware attribution for diagnostic insights. For YOLO-based car detection, the spurious attributions are spread more uniformly across all layers, unlike the eyeglasses task where they are confined to coarse-level features

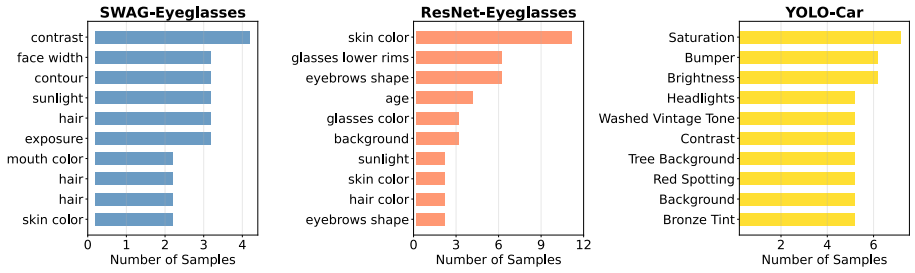

Fig. 9. RQ_3: Top-10 spurious features identified for SWAG, ResNet, and YOLO

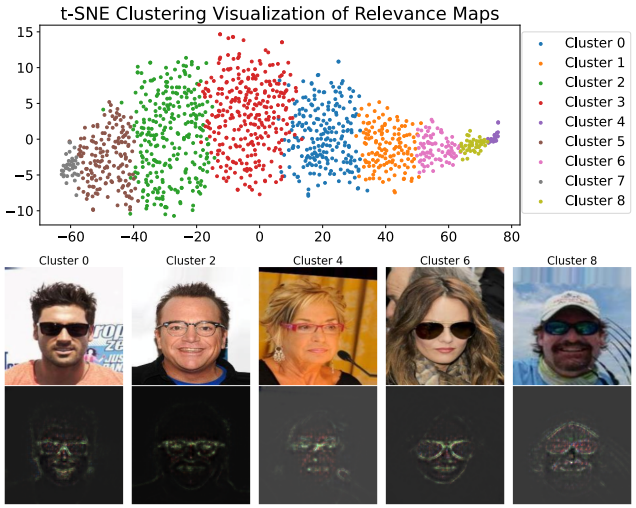

Fig. 10. RQ_3: t-SNE visualization of relevance maps by SpRAy. Bottom: samples from five distinct clusters, including original images and corresponding relevance maps.

Fig. 9 further reveals that SWAG's top spurious directions often reflect global appearance cues, such as face-width, contour, lighting intensity, and exposure. ResNet on the other hand has semantically finer spurious directions, such as glasses frame style, eyebrow height, or clothing. This contrast suggests that ResNet latches onto plausible but misleading features, while SWAG responds more to structural or environmental noise.For YOLO car detection, the spurious features span a broader range, including shifts in color saturation, bronze tint, contrast, which indicates that the car detector may rely on a wider mix of context signals than the attribute classifiers.

Finally, Fig. 10 shows a t-SNE of relevance maps clustered using Spectral Relevance Analysis (SpRAy). While the eigengap analysis suggests 9 clusters, the visualization shows a mostly continuous distribution. Cluster inspection reveals repeated attention around eyeglasses, raising ambiguity over whether these activations reflect causal semantics or spurious correlations. This highlights the limitations of pixel-level relevance methods and the strength of our feature-space probing in uncovering subtle spurious patterns.

RQ_3 (spurious features detection): *DETECT reveals substantial differences in spurious feature reliance between models with similar accuracy, showing that SWAG ViT focuses more on low-level, global cues while ResNet attends to finer but still irrelevant traits. In contrast, SpRAy fails to reliably identify such spurious patterns, especially when irrelevant features co-occur with class semantics.*

5.5.4 RQ_4 (model repair). On the CelebA test set, the fine-tuned model maintains performance comparable to the original model (target accuracy remains 0.997), indicating that tuning on repaired samples does not degrade performance on standard data. On the generated boundary-level test set, the improvements are clear: target accuracy increases from 0.935 to 1.000. These results show that the repaired dataset helps the model handle relevant feature perturbations more reliably while preserving its original behavior.

5.6 Threats to Validity

5.6.1 Internal validity. The performance of DETECT depends on the quality of the pre-trained StyleGAN generator and the accuracy of the semantic segmentation used in feature attribution. Errors in feature relevance estimation or latent disentanglement could lead to incorrect categorization of features as task-relevant or spurious. To mitigate this, we rely on pre-trained models when possible, and apply fine-tuning only when unavoidable, following established guidelines [23]. Additionally, the threshold selection for sensitivity and logit change might affect the precision of behavioral attribution.

5.6.2 External validity. Our evaluation is limited to specific datasets (*CelebA*, *COCO-Cars*, *Dogs*) and tasks (classification and detection). While we fine-tuned StyleGAN models for better alignment, results may not generalize to domains with less disentangled latent spaces or less structured visual features. Moreover, results on ResNet and SWAG may not extend to all model architectures or training regimes.

5.6.3 Reproducibility. All software artifacts and the complete results set are available in our replication package [1].

6 DISCUSSION

6.1 From Class- to Feature-Level Testing

Most prior DL testing methods lack semantic control, making test failures hard to interpret [20, 39]. MIMICRY introduced class-conditional testing, improving usability but still limited by the coarser $\mathcal{W}$-space. In contrast, DETECT operates in $\mathcal{S}$-space and targets specific semantic features based on model behavior, enabling fine-grained objectives. As shown in RQ_2, DETECT achieves better boundary discovery on attribute-driven tasks like "glasses detection" with higher MS-SSIM and lower d_2 distances than MIMICRY, indicating subtler, more targeted manipulations. On the dog dataset, both tools perform similarly, though DETECT preserves structure better. In the car detection task, DETECT is less effective, likely due to the robustness of the object detector requiring coarse-level changes.

A benefit of DETECT is its compatibility with unconditional generators as it directly perturbs feature dimensions without needing origin-target class seeds, unlike MIMICRY. This quality aspect of DETECT is also related to its limitation: selecting only task-relevant channels improves boundary detection for localized attributes (e.g., glasses),

but may affect effectiveness on broader concepts like gender, which require more global semantic variation.

6.2 Spurious Detection vs. Classifier Quality

The utility of spurious feature detection is closely tied to classifier quality. For low-performing models, such as in fine-grained multi-class dog breed classification, misclassifications frequently stem from underfitting or class confusion rather than shortcut reliance, making spurious attribution less meaningful. For these models, test generation is better viewed as a conventional generalization tool. Our framework supports this mode naturally by bypassing feature relevance identification with the more broadly used misclassification oracle.

Conversely, for high-performing classifiers, accuracy is not sufficient to assess reliability. As shown in RQ_3, both ResNet and SWAG achieve strong accuracy on the glasses task, yet our analysis reveals they rely on different features, including quite distinct spurious patterns. These findings reinforce the need for feature-aware testing that distinguishes causal semantics from confounding correlations, offering diagnostic insights that go beyond accuracy metrics. To this aim, the proposed metric relevance-based ratio provides a complementary perspective to conventional accuracy.

6.3 Explainability in Test Generation

Incorporating feature-aware perturbations provides a structured form of explainability. Rather than blindly searching for misclassification like Mimicry, Detect distinguishes between task-relevant and irrelevant features, enabling a more informative testing process and more precise judgments about model behavior: failures triggered by irrelevant features reveal spurious reliance (i.e., lack of robustness), while those triggered by relevant ones indicate a lack of generalization.

This fine-grained analysis is particularly important for high-accuracy models, where most inputs are correctly classified. We complement it with additional insights on *why* they are correct. Our framework supports this by converting test generation into a targeted probe of the model's causal reasoning. This not only improves test quality but also enables practical feedback for model assessment, debugging, and retraining.

6.4 Latent-Space Interpretability

Compared to conventional saliency-based methods, our framework introduces a more structured form of explainability grounded in latent space semantics. As shown in RQ_3, methods like SpRAy produce pixel-level heatmaps that highlight the part where the model focuses, but offer limited insight into what types of features affect the prediction. In contrast, Detect explains model behavior in terms of interpretable latent factors, offering concept-level attribution rather than low-level localization. Detect bridges the gap between interpretability and explainability: it does not merely visualize what the model reacts to, but enables the structured reasoning why it reacts that way.

7 RELATED WORK

7.1 Test Generation for DNNs

There are several strategies for generating test inputs for dc learning systems. In this section, we focus on three dominant approaches and contrast them with the capabilities of DETECT.

Raw Input Manipulation. These methods work directly on the input space, often by perturbing pixel values to induce misbehavior [9, 14, 28, 36, 45]. Common in adversarial and robustness testing, these techniques apply noise, occlusions, or transformations to existing inputs. While effective, they often produce unrealistic artifacts and are limited in exploring novel input variations [40]. Since the semantic attributes of the image usually remain unchanged, these methods offer limited insight into how the SUT reacts when meaningful features vary.

Model-based Techniques. They generate test inputs by manipulating high-level, structured representations of the input domain, such as parameterized models encoding domain constraints [12, 34, 39]. These methods enable semantically meaningful input synthesis by altering the model and converting it back to raw inputs. However, they rely on manually crafted domain models, which can limit their generalizability [40].

Generative AI-based Techniques. They have recently gained attention for their ability to produce structurally novel test cases without domain models. While Variational Autoencoders (VAEs) have been used [20, 30], they struggle with complex, context-rich image data [49]. GANs offer improved generative power [11, 30, 49], and recent work is beginning to explore diffusion models as well [30, 33]. Yet, they usually explore the latent space without distinguishing semantic features. The search in latent space often follows directions that mix attributes or rely on interpolation paths that drift toward invalid regions of the latent manifold, producing artifacts.

DETECT uses a generative AI-based technique, leveraging StyleGAN's architecture for fine-grained control over image features—a capability not yet explored in prior work. It integrates spurious feature detection into the generation process to increase the effectiveness of test cases.

7.2 Spurious Correlation Detection

Feature invariance is a foundational principle in robust and fair machine learning, which holds that a model's output should remain unchanged under variations in input features that are semantically irrelevant to the prediction task. This concept has been extensively studied in the context of invariant representation learning and fairness, where models are encouraged to ignore spurious attributes such as background, color, or sensitive demographic variables.

A prominent line of work is *Invariant Risk Minimization* [5], which aims to learn feature representations that support invariant optimal predictors across different environments. Similarly, methods in fairness-aware learning propose counterfactual tests to ensure that models do not change their predictions when protected attributes

are altered [24]. In the context of vision models, feature invariance has been used to detect model biases and to promote robustness against spurious correlations [3].

Our work builds on this intuition by introducing a test oracle that explicitly evaluates whether a model exhibits invariant behavior with respect to semantically irrelevant features. Unlike prior work that enforces feature invariance during training, our approach uses feature-level perturbations and behavioral analysis to uncover invariance violations in post-hoc testing and diagnosis.

8 CONCLUSIONS

We presented DETECT, a feature-aware test generation framework that bridges the gap between test generation and spurious feature diagnosis. By operating in a semantically disentangled latent space, DETECT enables fine-grained perturbations aligned with task semantics, and distinguishes between relevant and spurious feature influences. Our experiments show that DETECT uncovers behaviorally significant vulnerabilities in high-accuracy models, providing insight into how different models rely on distinct types of features.

As part of our future work, we plan to extend the framework beyond StyleGAN to other generative architectures, investigating how to effectively disentangle—or exploit existing disentangled—latent spaces for feature-aware testing. Another direction is extending the approach to modalities outside vision, such as natural language or audio, where latent structure and semantic drift present related challenges. These directions could broaden the applicability of DETECT to a wider range of DL testing.

REFERENCES

[1] 2025. Replication Package. https://github.com/Keith-XC/DETECT/.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[3] Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Jean-Christophe Gagnon-Audet, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish. 2021. Invariance principle meets information bottleneck for out-of-distribution generalization. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 263, 13 pages.

[4] Christopher J Anders, Leander Weber, David Neumann, Wojciech Samek, Klaus-Robert Müller, and Sebastian Lapuschkin. 2022. Finding and removing clever hans: Using explanation methods to debug and improve deep models. *Information Fusion* 77 (2022), 261–295.

[5] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893* (2019).

[6] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10, 7 (2015), e0130140.

[7] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923* (2025).

[8] Jacob Cohen. 1988. *Statistical power analysis for the behavioral sciences*. L. Erlbaum Associates.

[9] Francesco Croce and Matthias Hein. 2020. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International conference on machine learning*. PMLR, 2196–2205.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[11] Swaroopa Dola, Rory McDaniel, Matthew B Dwyer, and Mary Lou Soffa. 2024. CIT4DNN: Generating Diverse and Rare Inputs for Neural Networks Using Latent Space Combinatorial Testing. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[12] Alessio Gambi, Marc Mueller, and Gordon Fraser. 2019. Automatically Testing Self-driving Cars with Search-based Procedural Content Generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Beijing, China) *(ISSTA 2019)*. ACM, New York, NY, USA, 318–328. https://doi.org/10.1145/3293882.3330566

[13] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence* 2, 11 (2020), 665–673.

[14] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. 2018. Dlfuzz: Differential fuzzing testing of deep learning systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 739–743.

[15] Dongyoon Han, Sangdoo Yun, Byeongho Heo, and YoungJoon Yoo. 2021. Rethinking Channel Dimensions for Efficient Model Design. arXiv:2007.00992 [cs.CV]

[16] Yujin Han and Difan Zou. 2024. Improving group robustness on spurious correlation requires preciser group inference. In *Proceedings of the 41st International Conference on Machine Learning* (Vienna, Austria) *(ICML'24)*. JMLR.org, Article 696, 25 pages.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. , 770–778 pages. https://doi.org/10.1109/CVPR.2016.90 arXiv:1512.03385 [cs.CV]

[18] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL] https://arxiv.org/abs/2310.06825

[19] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. 2023. *Ultralytics YOLO*. https://ultralytics.com If you use this software, please cite it using this metadata. Released under AGPL-3.0. Repository: https://github.com/ultralytics/ultralytics.

[20] Sungmin Kang, Robert Feldt, and Shin Yoo. 2020. Sinvad: Search-based image space navigation for dnn image classifier test input generation. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 521–528.

[21] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2021. Alias-Free Generative Adversarial Networks. In *Proc. NeurIPS*.

[22] Tero Karras, Samuli Laine, and Timo Aila. 2021. A Style-Based Generator Architecture for Generative Adversarial Networks . *IEEE Transactions on Pattern Analysis & Machine Intelligence* 43, 12 (Dec. 2021), 4217–4228. https://doi.org/10.1109/TPAMI.2020.2970919

[23] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and Improving the Image Quality of StyleGAN . In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 8107–8116. https://doi.org/10.1109/CVPR42600.2020.00813

[24] Matt Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual fairness. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 4069–4079.

[25] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller. 2019. Unmasking Clever Hans predictors and assessing what machines really learn. https://doi.org/10.1038/s41467-019-08987-4

[26] RoboFlow User ldrago. 2022. Dogs and Cats Classifier Dataset. Kaggle. https://www.kaggle.com/datasets/rajarshi2712/dogs-and-cats-classifier

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*. Springer, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

[28] Yue Liu, Lichao Feng, Xingya Wang, and Shiyu Zhang. 2022. DeepBoundary: A Coverage Testing Method of Deep Learning Software based on Decision Boundary Representation. In *2022 IEEE 22nd*

International Conference on Software Quality, Reliability, and Security Companion (QRS-C). IEEE, 166–172.

[29] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[30] Maryam, Matteo Biagiola, Andrea Stocco, and Vincenzo Riccio. 2025. Benchmarking Generative AI Models for Deep Learning Test Input Generation. In *Proceedings of 18th IEEE International Conference on Software Testing, Verification and Validation (ICST '25)*. IEEE, 12 pages.

[31] Raghav Mehta, Vítor Albiero, Li Chen, Ivan Evtimov, Tamar Glaser, Zhiheng Li, and Tal Hassner. 2022. You Only Need a Good Embeddings Extractor to Fix Spurious Correlations. https://doi.org/10.48550/arXiv.2212.06254

[32] K. W. Morton and D. F. Mayers. 2005. *Numerical Solution of Partial Differential Equations: An Introduction*. Cambridge University Press, USA.

[33] Nusrat Jahan Mozumder, Felipe Toledo, Swaroopa Dola, and Matthew B. Dwyer. 2025. RBT4DNN: Requirements-based Testing of Neural Networks. arXiv:2504.02737 [cs.SE] https://arxiv.org/abs/2504.02737

[34] Neelofar Neelofar and Aldeida Aleti. 2024. Identifying and Explaining Safety-critical Scenarios for Autonomous Vehicles via Key Features. *ACM Transactions on Software Engineering and Methodology* 33, 4 (2024), 1–32.

[35] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: analysis and an algorithm. In *Proceedings of the 15th International Conference on Neural Information Processing Systems: Natural and Synthetic* (Vancouver, British Columbia, Canada) *(NIPS'01)*. MIT Press, Cambridge, MA, USA, 849–856.

[36] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated whitebox testing of deep learning systems, In proceedings of the 26th Symposium on Operating Systems Principles (Shanghai, China). *Commun. ACM* 62, 11, 1–18. https://doi.org/10.1145/3132747.3132785

[37] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. 2020. Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering* 25 (2020), 5193–5254.

[38] Vincenzo Riccio and Paolo Tonella. 2020. Model-based exploration of the frontier of behaviours for deep learning system testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 876–888.

[39] Vincenzo Riccio and Paolo Tonella. 2020. Model-based exploration of the frontier of behaviours for deep learning system testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 876–888.

[40] Vincenzo Riccio and Paolo Tonella. 2023. When and Why Test Generators for Deep Learning Produce Invalid Inputs: an Empirical Study. In *Proceedings of 45th International Conference on Software Engineering (ICSE '23)*. ACM, 12 pages.

[41] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 618–626. https://doi.org/10.1109/ICCV.2017.74

[42] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

[43] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017).

[44] Alexander Sorokin and David Forsyth. 2008. Utility data annotation with amazon mechanical turk. In *2008 IEEE computer society conference on computer vision and pattern recognition workshops*. IEEE, 1–8.

[45] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. arXiv:1312.6199 [cs.CV] https://arxiv.org/abs/1312.6199

[46] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

[47] V. Vapnik. 1991. Principles of risk minimization for learning theory. In *Proceedings of the 5th International Conference on Neural Information Processing Systems* (Denver, Colorado) *(NIPS'91)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 831–838.

[48] Z. Wang, E.P. Simoncelli, and A.C. Bovik. 2003. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Vol. 2. 1398–1402 Vol.2. https://doi.org/10.1109/ACSSC.2003.1292216

[49] Oliver Weißl, Amr Abdellatif, Xingcheng Chen, Giorgi Merabishvili, Vincenzo Riccio, Severin Kacianka, and Andrea Stocco. 2025. Targeted Deep Learning System Boundary Testing. arXiv:2408.06258 [cs.SE] https://arxiv.org/abs/2408.06258

[50] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (Dec. 1945), 80. https://doi.org/10.2307/3001968

[51] Zongze Wu, Dani Lischinski, and Eli Shechtman. 2021. StyleSpace Analysis: Disentangled Controls for StyleGAN Image Generation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12858–12867. https://doi.org/10.1109/CVPR46437.2021.01267

[52] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. 2015. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *arXiv preprint arXiv:1506.03365* (2015).

[53] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* 48, 1 (2020), 1–36. https://doi.org/10.1109/TSE.2019.2962027

[54] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2021. Deephyperion: exploring the feature space of deep learning-based systems through illumination search. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21)*. Association for Computing Machinery, 79–90.

A GENERATED EXAMPLES

(a) *CelebA*-glasses Task: Change whether Glasses are worn.

(b) *CelebA*-gender Task: Change Gender of the Person.

(c) *COCO-Cars* Task: Make YOLO see anything, but a Car.

(d) *Dogs* Task: Change the predicted Dog Breed.

Fig. 11. Origin ⇝ Perturbed Image Examples for DETECT and MIMICRY on each task.

Fig. 11 presents representative example pairs comparing DETECT and MIMICRY. Across tasks, DETECT produces more localized and attribute-specific perturbations, whereas MIMICRY often alters multiple features and introduces additional unintended changes. In general, perturbations generated by DETECT remain more controlled, modifying only the minimal region required for class transition. These examples illustrate the qualitative differences underlying the quantitative trends reported in the main text.

B FULL PROMPT EXAMPLES USED FOR VLM EVALUATION

The following prompts were examples that used for the VLM-based relevance attribution and relabeling for eyeglasses detection task.

Prompt for relevance attribution.

```
You are given three AI-generated images:

The first is the original face.
The second is generated by changing one latent direction.
The third is a difference mask showing where the change occurred.

Focus specifically on whether the presence of eyeglasses changes between
the first and second images.
Outout a JSON object with the following keys:
    {"answer": "yes, eyeglasses are added/removed"
       or "no, eyeglasses remain the same."}

Use the mask to support your reasoning and ignore irrelevant changes
such as lighting or color tone.
```

Prompt for relabeling.

```
You are an image tagger for AI-generated portraits.
For each image, output a JSON object with the following fields:
Determine whether the person in the image is wearing eyeglasses.
output a JSON object.
{ "glasses": "Yes / No / Ambiguous"}
If the image quality is poor or unclear, respond with "Ambiguous".
```